CPP 程式設計題

命題者：Chin-Fu , Yang

題目名稱(中文/英文)：ATM

主要測試觀念： Polymorphism

Basics		Functions	
☐	C++ BASICS 1	☐	SEPARATE COMPILATION AND NAMESPACES
☐	FLOW OF CONTROL	☐	STREAMS AND FILE I/O
☐	FUNCTION BASICS	☐	RECURSION
☐	PARAMETERS AND OVERLOADING	☐	INHERITANCE
☐	ARRAYS	☐	POLYMORPHISM AND VIRTUAL FUNCTIONS
☐	STRUCTURES AND CLASSES	☐	TEMPLATES
☐	CONSTRUCTORS AND OTHER TOOLS	☐	LINKED DATA STRUCTURES
☐	OPERATOR OVERLOADING, FRIENDS, AND REFERENCES	■	EXCEPTION HANDLING
☐	STRINGS	☐	STANDARD TEMPLATE LIBRARY
☐	POINTERS AND DYNAMIC ARRAYS	☐	PATTERNS AND UML

題目說明：A function that returns a special error code is usually better accomplished throwing an exception instead. The following class maintains an account balance.

```
class Account
{
private:
	double balance;
public:
	Account()
	{
		balance = 0;
	}
	Account(double initialDeposit)
	{
		balance = initialDeposit;
	}
	double getBalance()
	{
		return balance;
	}

	//returns new balance or -1 if error
	double deposit(double amount)
	{
		if (amount > 0)
			balance += amount;
		else
			return -1;
```

```
        return balance;
    }

    //return new balance or -1 if invalid amount
    double withdraw(double amount)
    {
        if ((amount > balance) || (amount < 0))
            return -1;
        else
            balance -= amount;
        return balance;
    }
};
```

Rewrite the class so that it throws appropriate exceptions instead of returning -1 as an error code. Write test code as shown in the following that attempts to withdraw and deposit invalid amounts and catches the exceptions that are thrown.

Note that please use this following code snippets as your main()

```
//Main
int main()
{
    Account a(100);
    try
    {
        cout << "Depositing 50" << endl;
        cout << "New balance: " << a.deposit(50) << endl;

        //cout << "Depositing -25" << endl;
        //cout << "New balance: " << a.deposit(-25) << endl;

        cout << "Withdraw 25" << endl;
        cout << "New balance: " << a.withdraw(25) << endl;

        cout << "Withdraw 250" << endl;
        cout << "New balance: " << a.withdraw(250) << endl;
    }
```

```
        catch (InsufficientFunds) // InsufficientFunds: a class name
        {
                cout << "Not enough money to withdraw that amount." << endl;
        }
        catch (NegativeDeposit) // NegativeDeposit: a class name
        {
                cout << "You may only deposit a positive amount." << endl;
        }

        cout << "Enter a character to exit" << endl;
        char wait;
        cin >> wait;
        return 0;
}
// note that
// class NegativeDeposit {…};
 // class InsufficientFunds {…};
```

輸入說明：

輸出說明：

IO 範例 ：

	Sample Input	Sample Output
第一組測資與輸出	input-main1.txt	output1.txt
第二組	input-main2.txt	output2.txt

附屬資料：

☑解答程式：ATM.cpp（檔名）

☑測試資料： input-main1.txt、output1.txt、input-main2.txt、output2.txt

■ 易，僅需用到基礎程式設計語法與結構

□ 中，需用到多項程式設計語法與結構

□難，需用到多項程式結構或較為複雜之資料型態或結構

解題時間：20 分鐘。

其他註記：